

## How to configure ESLint for React Projects ?

Last Updated : 22 Jul, 2024

- 
- 
- 

ESLint in React is a JavaScript linting tool that is used for automatically detecting incorrect patterns found in ECMAScript/JavaScript code. It is used with the purpose of improving code quality, making code more consistent, and avoiding bugs.

### Prerequisites to configure Eslint in React Projects

- [React JS](#)
- [ESLint – Pluggable JavaScript linter](#)

### Steps to create the React application:

#### Step 1: Create React Project

```
npx create-react-app myreactapp
```

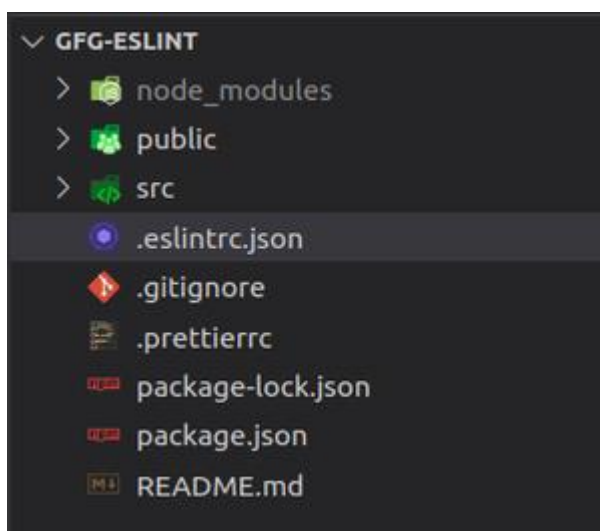
#### Step 2: Change your directory and enter your main folder

```
cd myreactapp
```

#### Step 3: Install ESLint in your React Project as a **devDependency** by running the following command

```
npm i -D eslint
```

### Project Structure



The updated dependencies in **package.json** file will look like:

```
{
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
```

```

    "@testing-library/user-event": "^13.5.0",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "devDependencies": {
    "eslint": "^8.51.0"
  }
}

```

### configure ESLint for React Projects

You can configure ESLint according to your use case. There are two ways to configure ESLint :

- **Configuration Comments:** These are JavaScript comments which are embedded into individual files to configure them
- **Configuration File:** ESLint will use JavaScript/JSON/YAML file which contain information to configure the entire directory.

In this particular config, we will use JSON format i.e. ``.eslintrc.json`` to have our configurations, or else you can create the ``.eslintConfig`` property in ``.package.json`` and write these configurations in that property.

#### Properties in ``.eslintrc.json``:

**“extends” “plugins”:** adding a file name in extends property we can inherit its configuration, whereas **plugin** works as an extension to ESLint which can perform numerous functions.

Inside our ``.eslintrc.json`` file add extends and plugin property similar to given below:

```

{
  "extends": [
    "eslint:recommended",
    "plugin:import/errors",
    "plugin:react/recommended",
    "plugin:jsx-a11y/recommended"
  ],
  "plugins": ["react", "import", "jsx-a11y"]
}

```

Note that as we have added various plugins we need to first install them so run the following command to install them as devDependencies :

```
npm install -D eslint-plugin-import eslint-plugin-jsx-a11y eslint-plugin-react
```

The ``.import-plugin`` will help us identify common problems while importing and exporting; ``.jsx-a11y`` will catch errors regarding accessibility and the ``.react`` plugin is all about code practices used in React, since we are using ``.eslint-plugin-react`` we will need to inform it which version of React we are using so let's add this in our **“settings”** property, instead of stating the current React version we will handover this job to settings by using the keyword **“detect”** so that it will detect the current React version from ``.package.json``

```

    ..},
    "settings": {
      "react": {
        "version": "detect"
      }
    }
  }
}

```

**“rules”**: Rules are used for configuring purposes, you can see all the rules that you can use <https://eslint.org/docs/rules/>. You can set the error level of rules in three different types :

- “off” or 0: This will turn off the rule.
- “warn” or 1: This will turn the rule on as a warning.
- “error” or 2: This will turn on the rule as an error.

Let’s add some rules to our config, you can add any other rules as per your choice from the list of all rules mentioned above.

```

"rules": {
  "react/prop-types": 0,
  "indent": ["error", 2],
  "linebreak-style": 1,
  "quotes": ["error", "double"]
},

```

**“env” and “parserOptions”**: In the “env” property, we will specify what environments we are working in. In parserOptions, we can specify JavaScript options like jsx support or ecma version

```

"parserOptions": {
  "ecmaVersion": 2021,
  "sourceType": "module",
  "ecmaFeatures": {
    "jsx": true
  }
},
"env": {
  "es6": true,
  "browser": true,
  "node": true
},

```

Final **.eslintrc.json** file

```

.eslintrc.json
{
  "extends": [
    "eslint:recommended",
    "plugin:import/errors",

```

```

    "plugin:react/recommended",
    "plugin:jsx-a11y/recommended"
  ],
  "plugins": ["react", "import", "jsx-a11y"],
  "rules": {
    "react/prop-types": 0,
    "indent": ["error", 2],
    "linebreak-style": 1,
    "quotes": ["error", "double"]
  },
  "parserOptions": {
    "ecmaVersion": 2021,
    "sourceType": "module",
    "ecmaFeatures": {
      "jsx": true
    }
  },
  "env": {
    "es6": true,
    "browser": true,
    "node": true
  },
  "settings": {
    "react": {
      "version": "detect"
    }
  }
}

```

Last but not least, let's add some commands in our package.json's "scripts" property to run ESLint

```

"scripts": {
  "lint": "eslint \"src/**/*.js,jsx\""
}

```

```
    "lint:fix": "eslint \"src/**/*.{js,jsx}\" --fix"
  },
```

Final **package.json** file

```
{
  "name": "gfg-eslint",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "react": "^18.3.1",
    "react-dom": "^18.3.1",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject",
    "lint": "eslint \"src/**/*.{js,jsx}\"",
    "lint:fix": "eslint \"src/**/*.{js,jsx}\" --fix"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "devDependencies": {
    "eslint": "^8.57.0",
    "eslint-plugin-import": "^2.29.1",
    "eslint-plugin-jsx-a11y": "^6.9.0",
    "eslint-plugin-react": "^7.35.0"
  }
}
```

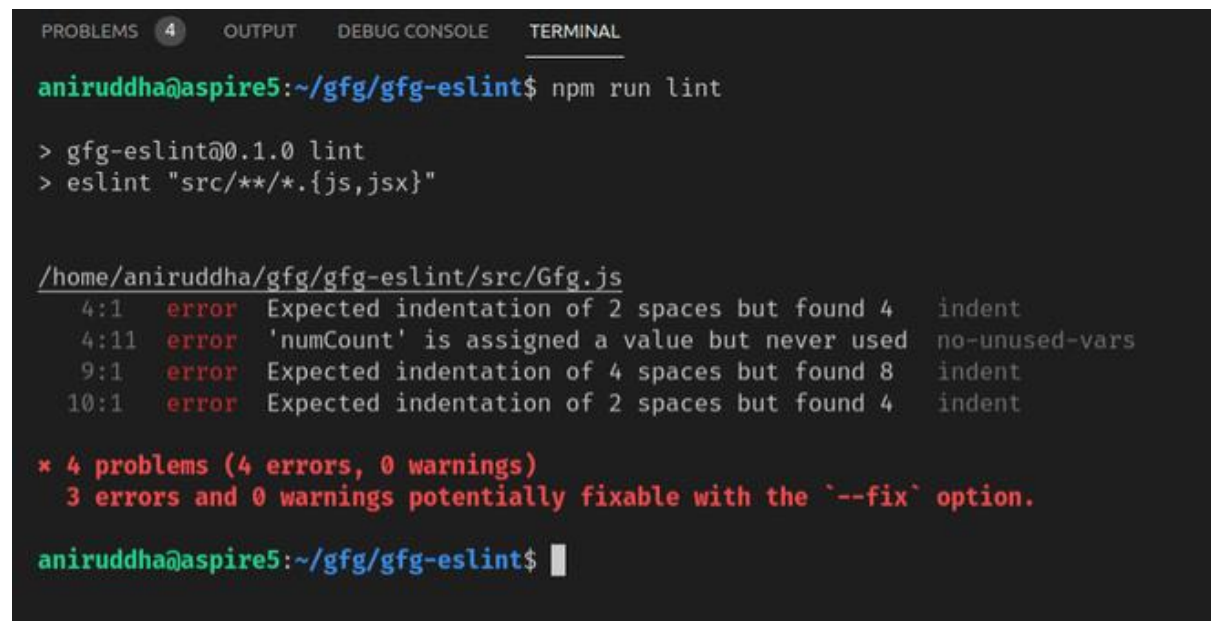
```
}  
}
```

The “lint” command will run ESLint inside every file in the “src/”, even if your “src/” folder contains multiple directories in it, this regex command will go down recursively on them and run ESLint; If some problems are reported by ESLint which are auto-fixable, then “lint:fix” command will do those auto-fixes.

**Step to run lint:** Open the terminal and type the following command.

npm run lint

**Output:** Output will be visible on the Terminal window as given below.



The screenshot shows a VS Code terminal window with the 'TERMINAL' tab selected. The terminal displays the command 'npm run lint' and its output. The output shows four errors related to indentation and an unused variable in the file '/home/aniruddha/gfg/gfg-eslint/src/Gfg.js'. The errors are: 'Expected indentation of 2 spaces but found 4' (twice), and ''numCount' is assigned a value but never used'. A summary at the bottom states '4 problems (4 errors, 0 warnings)' and '3 errors and 0 warnings potentially fixable with the --fix option'.

```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL  
aniruddha@aspire5:~/gfg/gfg-eslint$ npm run lint  
  
> gfg-eslint@0.1.0 lint  
> eslint "src/**/*.{js,jsx}"  
  
/home/aniruddha/gfg/gfg-eslint/src/Gfg.js  
  4:1   error  Expected indentation of 2 spaces but found 4  indent  
  4:11  error  'numCount' is assigned a value but never used no-unused-vars  
  9:1   error  Expected indentation of 4 spaces but found 8  indent  
 10:1  error  Expected indentation of 2 spaces but found 4  indent  
  
* 4 problems (4 errors, 0 warnings)  
  3 errors and 0 warnings potentially fixable with the `--fix` option.  
  
aniruddha@aspire5:~/gfg/gfg-eslint$
```